DOCUMENT RESUME

ED 284 536 IR 012 779

AUTHOR Curley, Wendy Paterson; Strickland, James
TITLE Garbage In/Garbage Out: Evaluating Computer Software.
PUB DATE 14 Apr 86
NOTE 16p.; Paper presented at the Annual Symposium of the New York College Learning Skills Association (Ellenville, NY, April 14, 1986).
PUB TYPE Guides - Non-Classroom Use (055) -- Speeches/Conference Papers (150) -- Tests/Evaluation Instruments (160)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Software; Educational Objectives; *Evaluation Criteria; Instructional Design; *Instructional Material Evaluation
IDENTIFIERS *Software Evaluation

ABSTRACT

Guidelines for understanding computer-assisted instruction (CAI) pedagogical designs and evaluating CAI software for its relevance to specific teaching and learning objectives are presented in this paper. Topics discussed include applications, drawbacks, and examples of the five types of CAI software--drill and practice, tutorial, simulation, problem solving, and educational games. An open-ended evaluation instrument which is simple to use and which emphasizes instructional objectives over methods is described, and a copy of the evaluation form and a list of recommended sources for software reviews are appended. (MES)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED284536

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Garbage In/Garbage Out: Evaluating Computer Software

A Paper Presented at the
New York College Learning Skills Association
(NYCSLA)
Annual Symposium
April 14, 1986
Ellenville, New York

Wendy Paterson Curley
Director of the Developmental Skills Center
Trocaire College, Buffalo, New York

James Strickland
Assistant Professor of English
Slippery Rock University of Pennsylvania

IR012779

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
James Strickland

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Garbage In/Garbage Out: Evaluating Computer Software

Wendy Paterson Curley
James Strickland

You just bought a computer or maybe your department was just given one. Perhaps you even came back to school this semester and found a roomful of computers. You may already know how to do word-processing, but you get the feeling there must be something more. Isn't there a whole world of software out there? You look in catalogues or visit computer stores and your head is spinning. Not only is there a ton of software out there, there's no way to tell what's good from what's bad. No wonder, as English teachers, we often feel frustrated when ordering computer-assisted instruction (CAI) software for our classrooms: frustrated in reviewing software that sounds good in a catalog but is disappointing in practice; frustrated in not knowing what to look for; frustrated in wanting to incorporate the new technology into our teaching but cautious in our acceptance of that technology. Our frustration with computer software in general comes from unrealistic expectations about what computers and computer-assisted instruction are capable of. We need to understand how the different CAI pedagogical designs are best used and how software should be evaluated for its relevance to specific teaching and learning objectives.

An appropriate expression in computer programming, "garbage in/garbage out," warns us that the results from the computer will only be as good as the input given. We can take that expression metaphorically that the results from using computer-assisted instruction in our classrooms will only be as

good as the input, the software, chosen to be available to our students. This presentation will:

--offer a brief explanation of each of the various types of software available;

--discuss some of beneficial applications, drawbacks, and examples of each type;

--and then offer an evaluation instrument that is simple to use, open-ended, grounded in needs analysis, and based on the question, "What do I want from computer-assisted instruction?" rather than the question, "How can I use this?"

Before looking at the five types of computer-assisted instruction, I would mention some reputable sources of software reviews, reviews which are written by English teachers for other English teachers, not reviews written to sell or promote software.

---

figure 1 about here

The five types of computer-assisted instruction are generally considered to be: Drill and Practice, Tutorial, Simulation, Problem Solving, and Educational Games.

Drill and Practice programs, the mostly widely known application of computers to instruction, are reductive, breaking down major concepts into skills discrete enough to be modularized and learned through repetition. As instructors, we generally drill on topics that require immediate feedback, such as mathematics (i.e. the multiplication tables) or factual

material (i.e. the Battle of Hastings--1066). If we believe there are skills which will benefit from a drill and practice presentation, then the computer has a remarkable ability to randomize and generate problems from a bank of questions. For example, a program called "Drill/Practice (And Instruction)" by Wendy Duignan of Niagara University, available from H & H Publishing, promises to "provide endless practice of the basic skills of arithmetic and beginning algebra." Unlike offering those same questions in a workbook or on a test, the computer can keep track of errors, report scores and error patterns, and repeat with further examples of problems answered incorrectly.

The Tutorial is an individualized presentation of course material already covered in class or new material better learned in a one-to-one setting, distinct from drill and practice's review of material previously presented. A computer tutorial must consider how much information to present at once and how to present it--both normal classroom considerations. Unlike activities planned for a classroom, computer tutorials must anticipate questions and offer a chances to ask questions. A good tutorial offers "help" screens, giving more explanation or further illustration to those who request it. It should come as no surprise that many computer-assisted tutorials are available to learn about computers, such as the "Introduction to Computers" series for the IBM, Apple, and TRS-80 computers by Steven Mandell, published by West Publishing. Often tutorials such as Mandell's rely on periodic "checkpoint questions" to insure the student is following the instruction. Wrong answers, in this case, are just as revealing as right answers. A good tutorial must be able to interpret errors and

continue the lesson from that point. For example, if an incorrect answer is chosen, indicating that the student has an incomplete understanding of the concept, the tutorial should branch to an explanation of that concept before continuing with the new information.

A Simulation program, the third type of computer-assisted instruction, creates a situation and allows students to act in the situation. A simulation can afford opportunities to test skills in real applications, for example, letting nursing students handle a patient, performing all the proper procedures and making decisions without risk to a "live" patient. A simulation can be used in applications as diverse as naval flight programs or car-design testing.

Our best CAI programs for writing use simulations, since writing skills are best learned in a simulated writing task, complete with full rhetorical setting. Rhetorical invention programs simulate the kinds of questions proficient writers ask themselves as they explore their topic. Audience awareness programs simulate the analysis of audience that proficient writers use to help them make strategic and lexical choices. Editing programs simulate the careful proofreading and global revision strategies good writers use. Beware, however, of writing programs, like Writing is Thinking by Kapstrom, which claim to simulate the process that good writers use, but, instead, work from an erroneous model that says first you form a thesis statement, then you make an outline, then you write some sentences about each item in your outline, and finally you check your writing against your outline to insure that you've covered everything and that your writing fulfills your thesis

statement. We know from protocol studies of real writers that almost no one goes through this procedure; therefore, this program does not model or simulate anything in the real world. Better programs are those like Writer's Helper by William Wresch, available from Conduit, and The Computer Resource Kit by Stephen Marcus, available from D. C. Heath. These programs offer a series of smaller programs to help students explore topics creatively and evaluate what they've written.

Another form of computer-assisted instruction is Problem solving, presenting a problem for students to solve using skills which they already possess. These programs are not examinations of, instruction in, or application of any specific content knowledge. They are, rather, an application of basic problem-solving strategies: means-end analysis, searching a problem space, brainstorming, heuristics, working backward, incubation. Visual Synectics, a program by Ray and Dawn Rodrigues, now of Colorado State University, applies to the writing process the problem-solving techniques called "synectics," developed for business and industry by William Gordon. Other types of computer-assisted instruction teach BASIC programming as a problem solving strategy. It becomes difficult to tell if the purpose of such a program is to teach content, a tutorial matter, or to teach flowcharting and procedure composition as a method of teaching problem-solving strategies.

The most difficult category to evaluate is Instructional or Educational Games. One program called the Grammar Examiner, published by Designware, promises an interactive board game where students advance from cub reporters to city editors on

the strength of their grammar skills. While grade school students seem to enjoy earning job promotions by avoiding comma splices and defending the galaxy by correctly adding a string of numbers, it is questionable whether the skills used in the game are transferred to other concepts. Often the game format is used simply as a motivational tool and makes the students go through elaborate game procedures to master a minimal number of skills.

Obviously, these categories have soft edges and deciding where to place any individual program is not as important as using the categories to help us decide whether a program fits our objectives. While all five types of programs contain an instructional component, only the tutorial is obviously instructional. Drill and practice corrects what has been learned, Simulation applies what has been learned, and Problem-Solving develops learning strategies.

THE EVALUATION FORM

When we began our search for the ultimate evaluation form, we found most, if not all, of the evaluation forms were checklist based. We found that checklists can cause evaluators to get so involved in categorizing whether the software is "this or that" on a "scale of 1 to 5" that they lose the essence of their subjective evaluation. When we look at a piece of software we know what we like and do not like--categorizing these feelings becomes an unnecessary chore for ourselves and of little value to someone else. We would rather read an instructor's subjective evaluation of a software

program's strengths and weaknesses in an open style rather a checklist, just as we would find an instructor's evaluation of a student's abilities much easier to understand if written in an open style.

---

figure 2 about here

Checklists also tend to be software oriented rather than instruction oriented. Figure 1 shows an evaluation form we feel will make the evaluation more relevant, a form designed not as a checklist but as a form emphasizing the priority of instructional objectives over methods. If we do not first formulate for ourselves how we will integrate the software into our course, we will focus too much on technical considerations, becoming carried away by slick packaging rather than sound pedagogy.

WHAT ARE YOUR OBJECTIVES?

The first question asks an instructor to clarify how the software will be used in the course. Will the software be used: to tutor? to give students the chance to manipulate information already presented in class? to give supplemental information? to provide drill and practice? to generate material for discussion in class? to give more examples of a particular concept? to give students a different medium for information exchange (other than reading and studying notes)? to correct errors in their writing? to give practice writing?

These questions help clarify which type of software, as

discussed above, is needed to fulfill course requirements. An instructor looking for a drill and practice program should not reject a program which lacks critical thinking skills. Another instructor looking for a program that will give students the opportunity to manipulate information and draw conclusions should not choose a "workbook on a screen" program. If the instructor's objective is to help students learn to correct errors in their writing, a program that spots possible errors in student-generated text is quite different from one which teaches error correction of programmed text. Unless an instructor is careful about stating objectives for the software, these distinctions in software might cause the instructor to fit the course to the software, instead of the other way around.

WHAT DO YOU HOPE THIS WILL DO FOR YOUR STUDENTS?

After using the software, will the students be able to: compose with better ideas? read their text better? write exams better? correct grammatical errors? understand concepts? understand details? manipulate information?

These questions help clarify what types of programs not to buy. If an instructor wants students to compose with better ideas, then a grammar drill, no matter how highly rated, will not help them. If an instructor wants students to respond to questions from their own manipulations of material, then software which uses multiple choice questions will not help them. And if an instructor wants students to get more information than what was presented in class, a tutorial review will not help them.

TECHNICAL CONSIDERATIONS

Instructors often shy away from technical matters, but some

considerations of a technical nature should be weighed. A technical question such as "Does the program present text a full screen at a time or word by word, line by line?" is an important one to ask since efficient readers do not necessarily track eye movements smoothly from left to right. If "user friendliness" is important, some technical conveniences are worth investigating: ease of moving backward, forward, and controlling screen-advance in a program; frame-by-frame help and escape options; ability to retain incomplete exercises; protection against destruction, or "system crashes," by a program "bug" or a student mistake; ease of correcting keystroke errors (backspace, arrows, control-key combination); independence from a printed manual.

INSTRUCTIONAL CONSIDERATIONS

Instructional considerations are the major part of an open-ended, holistic evaluation of software. Software produced by persons other than educators sometimes does not show a deep understanding of all the subskills involved in analyzing the material presented. Programs may use a reductive approach (skills oriented) to teach a skill which really requires a holistic approach, a demonstration of all the subskills acting as a whole, not acting independently. This reductive approach is one explanation why pre/post test scores, used to prove effectiveness of software, look so good but do not measure the degree to which a student is able to apply those skills to coursework.

Software produced by publishers is also promoted by publishers. Publishers pay close attention to the "buzz" words (or "god" words) of each discipline. Instructors need to ask

if publishers' terms like "interactive" and "process oriented" mean what they they think by them, or if the words are merely used to "sell" software.

A good course of instruction acknowledges the need to develop critical thinking and good teaching reflects this, but is it fair to expect CAI to do what even some good teachers cannot? If we evaluate all software for a critical thinking component, we will end up buying nothing. Although we do not want software to program students into a "push button" mentality, we must return to the primary question, "What are my objectives?" A content tutorial disk, even deficient in critical thinking strategies, can be a valuable part of a well-planned course, a program that incorporates critical thinking in its total structure.

---

figure 3 about here

A STUDENT EVALUATION

Finally, software must be evaluated by those who will use it--the students. If software can be borrowed or obtained on a preview basis, instructors should ask students from an appropriate course level to test the program. Figure 2 is a sample student evaluation form, asking some basic questions about the level of difficulty and the "user-friendliness" of the program but focusing on a written declaration of the program's best and worst features. The instructor's evaluation together with student evaluations will give a good indication

of how successful the computer-assisted instruction will be in a specific course. There is no perfect software; CAI must be evaluated in terms of the instructor's course objectives.

In conclusion, we are both encouraged by and disappointed in the CAI software presently available. Our only blanket statement would be that software produced by an instructor for a specific course, even with technical flaws, is by far superior to "canned" material. However, most of us lack the expertise to do all the good things we expect from professional publishers. Therefore, we must intelligently view, evaluate, and select published software to spend grant money on to enhance our programs, without succumbing either to overzealous purchase or overcritical rejection.

figure1:

LANGUAGE ARTS SOFTWARE REVIEWS YOU CAN TRUST

Research in Word Processing Newsletter
published monthly during academic year
$15.00 per year

Bradford Morgan & James Schwartz, Editors
South Dakota School of Mines & Technology
Rapid City, SD 57701-3995
605-394-2481

Computers and Composition
regular software review articles by Craig Waddell
three issues per year
$8.00 per year

Cindy Selfe & Kate Kiefer, Editors
Michigan Technological University
Houghton, MI 49931

English Journal
an NCTE journal
published monthly during academic year
regular software review articles by Bruce Appleby

Language Arts
an NCTE journal
published monthly during academic year
regular software review articles by Judith Newman

ACE Newsletter
four issues a year
$10 per year

Assembly on Computers in English
Tom Decker, Managing Editor
Westview Centennial Secondary School
755 Oakdale Road
North York, Ontario, Canada M3N 1W7

McDaniel, E. (1985). A bibliography of text-analysis and writing instruction software. Philadelphia: Temple University Working Papers in Composition. ($1.90) Also featured in Fall 1986 issue of Journal of Advanced Composition.

Schwartz, H. J., & Bridwell, L. S. (1984). A selected bibliography on computers in composition. College Composition and Communication, 35, 71-77.

figure 2:

SOFTWARE EVALUATION FORM

TITLE _____

PUBLISHER _____

_____

COURSE INTENDED FOR _____

LEVEL INTENDED FOR _____

WILL IT RUN ON THE COMPUTERS YOU HAVE? _____

WHAT ARE YOUR OBJECTIVES FOR USING THE SOFTWARE?

WHAT DO YOU HOPE THE SOFTWARE WILL DO FOR YOUR STUDENTS?

COMMENTS ON FORMAT:
HOW IS THE MATERIAL PRESENTED?

NOTES ON TECHNICAL CONSIDERATIONS:

NOTES ON INSTRUCTIONAL CONSIDERATIONS:

figure 3:

STUDENT EVALUATION FORM

Name of program you used _____

How many used it with you? _____

Did you think this program was __too easy __just right __too hard?

Were you able to read and understand the directions? _____

Did you have to ask the teacher or another student for help?_____

Did the computer help with incorrect answers?_____

Describe what you liked best about program:

_____

_____

_____

_____

Describe what you liked least about program:

_____

_____

_____

_____

Would you use this program again? _____

Would you recommend this program to another student? _____
If yes, why?

If no, why not?